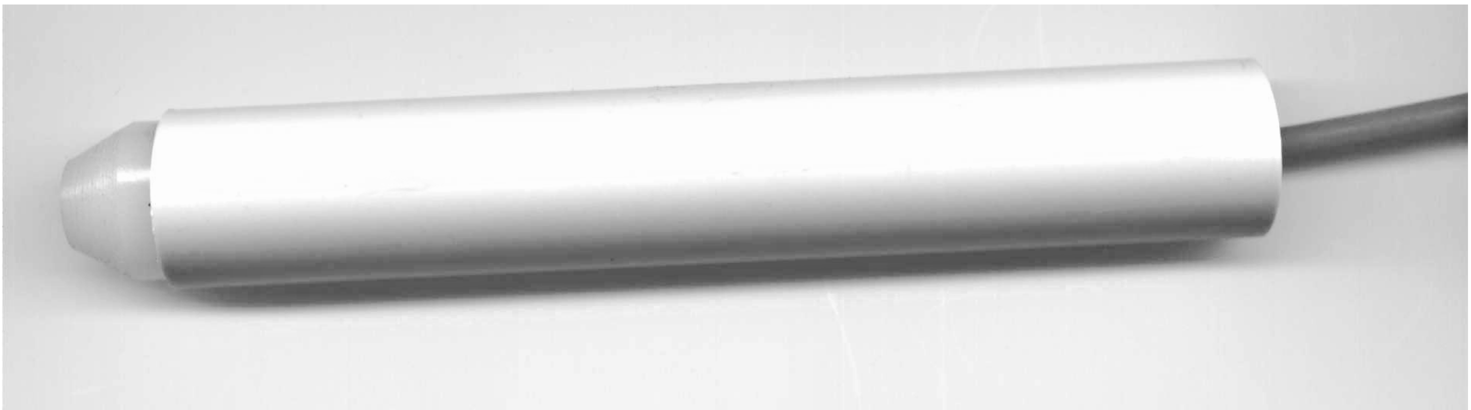
If a PIC is used for example in a timer application, such as a programmable clock timer, the majority of the circuit is taken up by the user interface with its switches, pushbuttons and display hardware. This expense can be avoided if the settings are entered via a PC, but operating the device is then made even more inconvenient by the required connecting cable. In this article, the author describes a simple interface that requires almost no additional hardware and allows a PIC application to be configured from the PC without a connecting cable.

Design by H. Bärnthaler

PC screen data flasher

a wireless interface between the PC and a PIC



As an example, let's suppose that you want to build a convenient timer using a microcontroller. All that you need for the timer function is a counter that engages or releases a relay at certain times. This function can be realized with a few lines of software.

However, you will need significantly more software and hardware (pushbuttons and a display) to be able to configure the timer. You must be able to set the current time and define the various switching times, you have to specify which relay is to be controlled, and of course you will have a choice of several options. For the benefit of the user, you compose four pages of verbose text in order to explain how to set up the timer using its three pushbuttons and three-character display.

A better solution

Instead of all this, you can write an easy-to-use C program on your PC, with attractive screen windows and a help menu, and use this to enter the necessary data for the timer. Then you hold a simple probe, resembling a ball-point pen and containing a phototransistor sensor, against a certain portion of the monitor screen where a blinking point of light appears. This transfers the required 20 bytes of configuration data to the microcontroller.

The advantages of this approach are obvious. Only one port lead is used for the entire key entry and configuration management functions of the timer. The display and pushbuttons are made fully redundant, since all data are entered at the PC and graphically edited. No cable need be connected to the PC, and the PC and the PIC system are fully electrically isolated. Finally, a small amount of simple code in the PIC is sufficient for handling the data, so that more memory is available for the actual application.

The configuration data are managed in a convenient and easily-understood manner using the PC, and the complete PC keyboard is effectively made available to the PIC. This procedure can be used with even the most antique PC, since practically every PC can display an alternating black and white rectangle on its monitor. Since none of the PC ports is used, the common problem of an incorrectly configured serial connection disappears entirely, and special PC hardware expertise (which is often needed with serial interfaces) is not required.

Of course, we should not neglect to mention the disadvantages: data can only be transferred from the PC to the PIC, and not in the other direction, and the data transfer rate is slow (around 40 bytes/minute). However, this is not a problem for many applications, where the amount of data to be transferred is anyhow very small.

How it works

A blinking rectangle is generated and displayed on the PC monitor, with pulse-width modulated brightness. This carries the data words in serial form. A phototransistor passes the signals to the microcontroller, which reconstructs the data words.

As can be seen from the schematic diagram in **Figure 1**, the necessary hardware is very simple. Any standard type of phototransistor can be used for the receiver; two common types are specified on the schematic. The phototransistor is followed by a single amplification stage using a BC548B, which increases the signal level so that it will reliably activate the Schmitt-trigger input of the following monostable. The pulses produced by the monostable are connected directly to the microcontroller port RB0 pin for evaluation within the PIC. The PIC shown in the sample circuit uses a 455-kHz ceramic resonator and is wired only to the extent necessary to test the interface program.

Software

A Pascal program that can be used for sending the data bytes from the PC is available on a diskette through the *Elektor Electronics* Readers Services, together with a commented assembly-language program listing for the PIC (order number 996038-1).

The sample program 'ScreenDataFlasher', shown in **Table 1**, demonstrates the function of the screen data interface and transfers a total of 20 bytes with alternating values of FF$_H$ and 00$_H$. An init bit is sent before each data byte; the microcontroller measures the length of this bit. The following data bits are pulse-width mod-

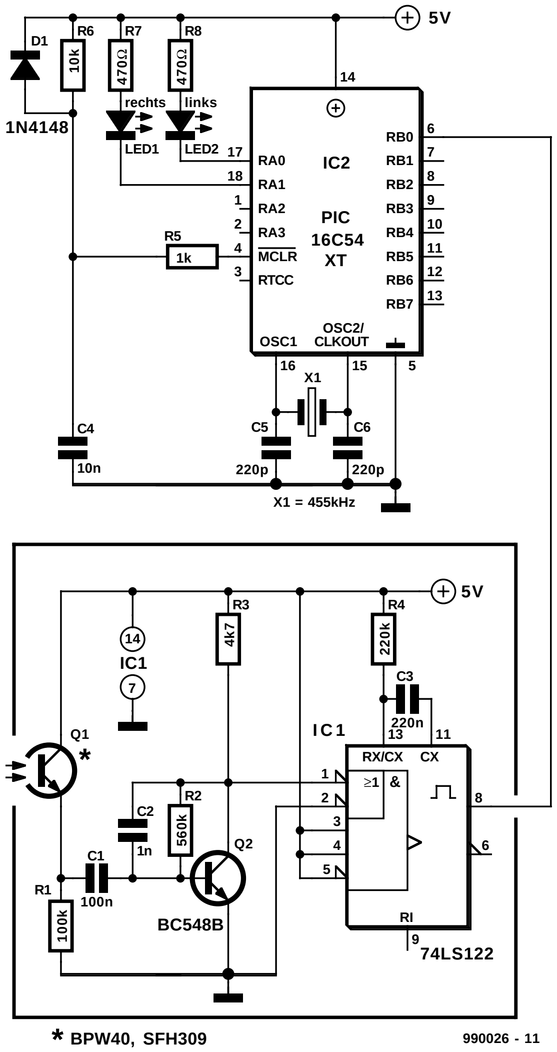


Figure 1. The hardware required for the interface is very simple, consisting of a phototransistor, an amplifier stage and a monostable.

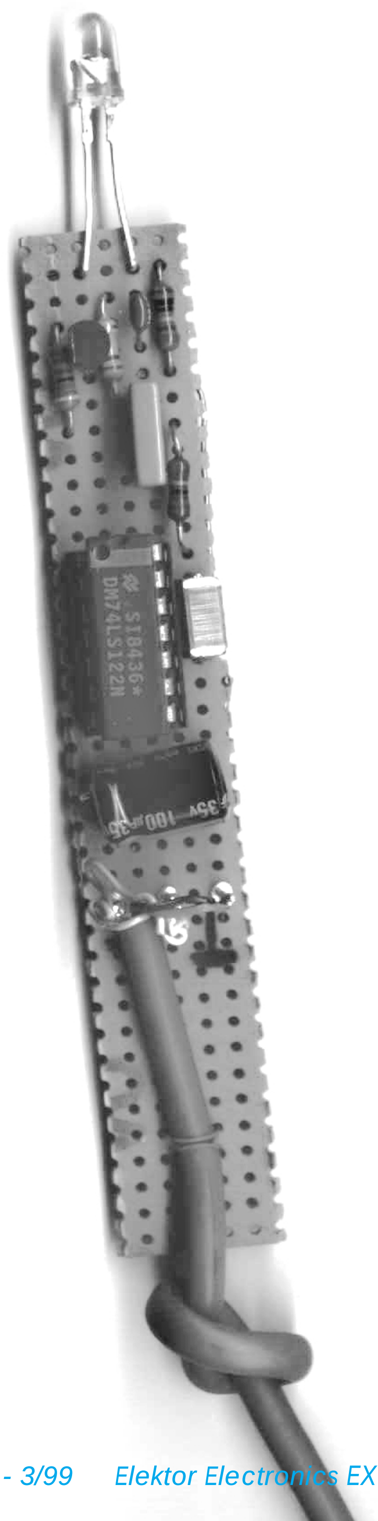
Table 1. Test and demonstration program for the interface.

```
Pascal Listing
************************************************************}
Program PCScreenDataFlasher (Input,Output);
Uses   Dos,CRT,Graph;

type
    PointType = record
            x,y: Word;
        end;

Var Var1             : Integer;
    Var2, Var3       : Integer;
    GraphMode, GraphDriver   : Integer;
    x1, y1, Width, Height    : Integer;
    MeasInstant      : Integer;

const
    Pentagon: array [1..4] of PointType =
    (( x: 50; y:50),
     ( x: 50; y:100),
     ( x: 100; y:100),
     ( x: 100; y:50));
    Dark: FillPatternType =  ($00, $00, $00, $00,
                                  $00, $00, $00, $00);
    Bright: FillPatternType =    ($FF, $FF, $FF, $FF,
                                  $FF, $FF, $FF, $FF);
    HighLength     = 2000;  {length of high pulse}
    LowLength      = 1000;  {length of low pulse}
```

ulated. All pulses that are longer that the init bit are decoded as ‘1’, and all pulses that are shorter than the init bit are decoded as ‘0’. The size of the data word is adjustable, with a minimum value of one bit. The actual setting must of course be taken into account in the microcontroller.

The specified time delays may have to be modified to accommodate the PC used and the monitor’s vertical repetition rate. Start testing with relatively long delay times and gradually reduce the delays until the data are no longer correctly transferred. If you have questions, you can contact the author via e-mail: Ing. H Bärnthaler at hbaernthaler@mail.carinthia.co.at.

(990026-1)



```
    WaitLength     = 1500;  {time between bits}
    InterByteTime = 1000;  {time between bytes}

{******************************************************************
{  In  this  subroutine  the  start  bit  is  flashed,  its  length  is
{ measured by the microcontroller.
******************************************************************}
Procedure InitBit;
Begin
    {transmit Init bit length calculation}
    MeasInstant := LowLength + ((HighLength-LowLength) div 2);
    SetFillPattern (Bright, White);
    FillPoly (SizeOf (Pentagon) div SizeOf(PointType), Pentagon);
    Delay (MeasInstant);
    SetFillPattern (Dark, Black);
    FillPoly (SizeOf (Pentagon) div SizeOf(PointType), Pentagon);
    Delay (WaitLength);
End;

{******************************************************************
 Program start
******************************************************************}
Begin
    GraphDriver := Detect;
    InitGraph (GraphDriver, GraphMode, ‘c:\pascal\graphix’);
    readln;  {Wait for key pressed}
    For Var1 := 1 to 10 do
        Begin
                InitBit;
                {supply 8 High Data bit}
                For Var2 := 1 to 8 do
                    Begin
                        {supply High}
                        SetFillPattern (Bright, White);
                        FillPoly (SizeOf (Pentagon) div
                        SizeOf(PointType), Pentagon);
                    Delay (HighLength);
                        SetFillPattern (Dark, Black);
                        FillPoly     (SizeOf     (Pentagon)     div
SizeOf(PointType), Pentagon);
                    Delay (WaitLength);
                   End;
         { wait until next data byte transmitted }
            Delay (InterByteTime);
            InitBit;
                {supply 8 Low Data Bit}
                For Var2 := 1 to 8 do
                    Begin
                        {supply Low}
                        SetFillPattern (Bright, White);
                        FillPoly     (SizeOf     (Pentagon)     div
SizeOf(PointType), Pentagon);
                    Delay (LowLength);
                        SetFillPattern (Dark, Black);
                    FillPoly      (SizeOf      (Pentagon)      div
SizeOf(PointType), Pentagon);
                    Delay (WaitLength);
                    End;
                {wait until next data byte is sent}
            Delay (InterByteTime);
        End;
         { display dark again}
    readln;
    CloseGraph;

End.
```